https://doi.org/10.2352/EI.2024.36.10.IPAS-248
© 2024, Society for Imaging Science and Technology

Examining the Effects of Compression on Deep Learning Remote Photoplethysmography

Benjamin Sporrer, Nathan Vance, Jeremy Speth, Patrick Flynn
University of Notre Dame
{bsporrer, nvance1, jspeth, flynn}@nd.edu

Abstract

Remote photoplethysmography (rPPG) is a camera based technique used to estimate a subject's heart rate from video. It exploits the difference in light reflection between blood-dense tissue and other tissue by detecting small variations in the color of RGB pixels on skin. While often undetectable to the human eye, these subtle changes are easily detectable from high-quality video. Working with high-quality video presents many challenges due to the amount of storage space required to house it, computing power needed to analyze it, and time required to transport it. These problems can be potentially mitigated through the use of compression algorithms, but modern compression algorithms are unconcerned with maintaining the small pixel intensity variations within or between frames that are needed for the rPPG algorithms to function. When provided with compressed videos, rPPG algorithms are therefore left with less information and may predict heart rates less accurately. We examine the effects of compression on deep learning rPPG with multiple commonly used and popular compression codecs (H.264, H.265, and VP9). This is done at a variety of rate factors and frame rates to determine a threshold for which compressed video still returns a valid heart rate. These compression techniques are applied against multiple public rPPG datasets (DDPM and PURE). We find that rPPG trained on lossless data effectively fails when evaluated on data compressed at compression constant rate factors (CRFs) of 22 and higher for both H.264 and H.265, and at a constant-quality mode of CRF above 37 for VP9. We find that training on compressed data yielded less accurate results than training on lossless or less compressed data. We did not find any specific benefit to training and testing on data compressed at identical compression levels.

Introduction

Remote photoplethysmography (rPPG) relies on small variations in RGB pixel intensity to signal the change in a subject's blood volume. These fine details are readily present in high-quality losslessly compressed videos, but such videos are very challenging to work with due to storage and computational costs. The ability to run rPPG algorithms on compressed videos is thus highly sought after [1, 2].

Modern video compression techniques typically maintain visual quality as perceived by humans and are therefore unconcerned with the loss of subtle information necessary for rPPG algorithms. In 2D image arrays, pixels are typically correlated spatially and information unnecessary for the human visual system is removed [3]. Since each picture is coded without reference to other pictures in the video sequence, this method of compression is known as intra coding. In video sequences, temporally correlated pixels also contain duplicated information [3, 4]. Employing compression techniques in the temporal domain is known as inter coding and is the key technique that separates video compression algorithms from standard image compression. By combining both intra and inter coding techniques, modern video compression algorithms are capable of maintaining visual quality with smaller videos files, thereby enabling easier storage, movement, and computation.

In order for rPPG algorithms to work effectively, they must be able to detect near imperceptible color changes that occur as blood passes under the skin. For individual pixels this change often happens at sub-noise levels, thus requiring a region of pixels to be aggregated [2]. By discarding redundant spatial or temporal information unneeded for visual fidelity, video compression algorithms inherently discard the small variations used by the rPPG algorithms. Even a small variation in the pixel information can undermine the ability of a rPPG algorithm to derive a signal from a specific region of pixels. As the amount of compression increases, rPPG algorithms have less detailed pixel information to work with and their ability to accurately predict heart rates begins to suffer.

This paper explores the effects of the popular H.264, H.265, and VP9 compression codecs on the ability of a temporally dilated 3DCNN architecture, RPNet [5], and a temporal shift convolutional attention network, TS-CAN [6], to produce accurate pulse waveforms. We perform these tests against two public rPPG specific datasets, DDPM [7] and PURE [8].

Related Work

During the early years of rPPG development, most techniques were tested on relatively small, private datasets. Researchers were more focused on generating and improving their own effective rPPG algorithms than on investigating potential pitfalls, such as the effects of compression. The development of public datasets enabled the comparison of techniques on a reasonable scale [5].

A seminal study on the effects of the H.264 and H.265 codecs on rPPG was first explored by McDuff et al. [2]. They utilized the constant rate factor (CRF), a factor that controls the adaptive quantization parameter to provide constant video quality, in an effort to understand the trade-offs between a video's visual fidelity, bit rate, and the accuracy of the cardiac frequency produced from rPPG [2]. The main conclusion of their work was that a considerable drop occurs in the signal-to-noise ratio (SNR) between compressed and raw videos. Špteli̊k et al. produced an analysis on the progress and limitations of rPPG [9]. They reported that the field was plagued by a lack of consistency in accurately representing compression formats as well as a vagueness in general compression terminology. A comprehensive study was performed by Rapczynski et al. [1] on two publicly available datasets, PURE [8] and MMSE-HR [10]. The videos within each database were compressed using the FFmpeg x264 and x265 implementations of H.264 and H.265 at every other CRF between 1 and 37. A CRF of 0 was chosen as the lossless baseline since it reduces data size by approximately 80% while still maintaining a strong PPG signal.

Nowara et al. [11] investigated the effects of training a deeply-recursive convolutional neural network (DRNN) tailored for recovering physiological information on compressed videos [12]. As in the initial compression study by McDuff et al. [2], the dataset of facial videos collected by Estepp et al. [13] was utilized. The attention based deep learning approach [12] was trained directly on compressed data and it was determined that there is significant advantage to training and testing on compressed video of a similar CRF. This result suggests that artifacts related to the compression algorithm may be learned and accounted for by the network's architecture. Thus, the best method for obtaining reliable rPPG from compressed data may be to have specialized networks for each level of possible compression.

In an effort to establish a generalized model Nowara et al. [14] performed a systematic test of the 12, 18, 24, 30, and 36 CRF levels across the H.264, H.265, and MPEG-4 compression standards. They concluded that a network trained on videos of the same or higher compression level can obtain reliable pulse waveforms from compressed videos. Models trained on compression levels between CRF12 and CRF24 are capable of generalizing within that range, with models trained on higher compression levels performing better.

Problem Definition

This paper explores how deep learning rPPG algorithms behave under three popular compression codecs, H.264, H.265, and VP9. The goal is to determine an acceptable level of compression that allows the rPPG techniques to still accurately predict a heartrate for each of the individual codecs. We compress rPPG specific datasets with frame rates of both 30 FPS and 90 FPS while varying the constant rate factor presets within FFmpeg.

We also explore how models trained on compressed data behave when tested on compressed data. This helps us to determine if training on compressed data is a viable strategy to pursue in balancing the need for accurate heart rates against storage and processing requirements.

Background

To test the effects of compression algorithms on rPPG algorithms, we examine three different compression codecs: H.264, H.265, and VP9. These three codecs were chosen because they have seen use in previous studies [14, 11, 1, 2, 9], are popular among the general public, and are likely to remain in use for several more years. We utilize the constant rate factor parameter for the codecs (called *constant quality* in VP9) in order to maintain the highest visual quality possible without limiting the exact bitrate or file size. Under this mode, videos are encoded with varying levels of compression in order to achieve a constant perceived visual quality across frames of varying complexity and motion [1, 2]. The compression algorithms are run with parameters as close to the default FFmpeg implementation settings as possible. The following sections provide background on compression, the codecs we use, RPNet, TS-CAN, and our datasets.

Video Compression

Compression algorithms are ubiquitous in the modern computing and internet landscape due to their ability to make an acceptable trade-off between bit rate, file size, and visual fidelity. In general, codec algorithms use a combination of temporal prediction between a sequence of video frames and spatial compression within individual frames to eliminate redundant information. This is done according to varying sets of parameters such that bit rates and file sizes can be reduced while still presenting acceptable visual quality.

Codecs

H.264

With a market share upwards of eighty-three percent, the most widely used and popular compression codec is H.264. For H.264 the constant rate factor (CRF) varies in range between 0 and 51, with lower values signifying that less compression is performed. For visual purposes, CRFs of 17 or 18 and below are considered to be essentially lossless. In typical compression, a range between 17 and 28 is consider to be acceptable with 23 being the selected default.

H.265

The H.265 codec was developed with the objective of providing increased compression efficiency over H.264. It provides 25-50% bitrate savings. The CRF functions identically to that of H.264, but produces visual equivalent compression at higher CRF values. The default CRF is 28 and should correspond to the H.264 default CRF of 23 in terms of visual quality while using only about half the file size.

VP9

The VP9 codec is an open-source video coding format that was developed by Google and competes with H.265. The default setting for VP9 is a two-pass encoding. We disabled the two-pass encoding mode in favor of the "Constant Quality" mode. This mode is similar to the CRF modes of H.264 and H.265. Despite being named differently the parameter is still denoted as CRF in the FFmpeg implementation and ranges from 0 and 63. The recommended range is between 15 and 35.

RPNet

We utilize our state of the art 3DCNN, RPNet, for rPPG analysis [5]. Our training regime seeks to minimize the negative Pearson correlation between the predicted waveform and ground truth waveform. As data progresses through RPNet's depth, the dilation rates it uses for its temporal convolutions increases, resulting in an increased receptive field. The dilated convolutions examine frames at the various past and future time points in order to produce an effective output [5]. The architecture is comprised of 3D convolutions, max-pooling along the spatial dimension, and global pooling across the feature maps [5].

Preprocessing

To prepare videos for RPNet [5], we use the MediaPipe Face Mesh tool [15] to crop the raw videos tightly around the face region. By using landmarks and cropping at the extreme points, irregularities in the bounding box are reduced. Bicubic interpolation is then applied to the cropped region in order to scale each frame in the video clip down to 64×64 pixels.

These frames are used as input for RPNet such that each input consists of 136-frame chunks. For every frame in a video, the model predicts a waveform value. A stride of 68 frames (half the length of the clip) is utilized between chunks. To produce a single waveform the outputs are normalized, a Hann window function is applied to the overlapping segments, and the results are added together.

To establish ground truth we utilize the CHROM [16] pulse detection technique to generate a face reference waveform that is capable of estimating the offset between the finger and face. We then calibrate the ground truth PPG signal by shifting it temporally such that the Pearson value is maximized between it and the CHROM waveform.

Postprocessing

Due to our evaluation methods requiring inferred heart rates, we take the Short-Time Fourier Transform (STFT) of the output waveform with a window size of 10 seconds and a stride of 1 frame. This creates a 10-second latency tolerance for our system in application scenarios. To reduce the quantization effects, the waveform is padded with zeros such that the bin width in the frequency domain is 0.001 Hz (0.06 bpm). After applying a bandpass filter at $.6\overline{6}$ and 3 Hz (40 and 180 BPM), we select the highest peak as the inferred heart rate [17].

TS-CAN

We investigate the effects of compression on the Temporal Shift Convolutional Attention Network (TS-CAN) [6]. This network was developed to investigate both the temporal and spatial features in a given video. Preprocessing of a video involves cropping around a subject's face and downsampling the resulting cropped video to 36 x 36 pixels. Pairwise differences between frames are calculated from the downsampled cropped video and fed into the network along with the raw frames in 20-frame segments.

To match both refinements for increased spatial resolution and the dimensions used with RPNet we increase the size of the input video to 64 x 64 frames [18]. The model layers are grouped following the rPPG-Toolbox implementation [19]. Postprocessing of the resulting waveform is performed via the method described in the "Postprocessing" section outlined above.

Metrics

The metrics used to evaluate our data are the same as the ones proposed by Speth et al. during the development of RPNet [5]. Each video has its metrics calculated independently.

Mean Error (ME)

The ME captures the bias of the method in BPM, and is defined as follows:

$$ME = \frac{1}{N}\sum_{i=1}^{N}(HR'_i - HR_i) \tag{1}$$

Where HR and HR' are the ground truth and predicted heart rates, respectively. Each contained index is the heart rate obtained from the STFT window as specified in the "Postprocessing" section, and N is the number of STFT windows present.

Mean Absolute Error (MAE)

The MAE captures an aspect of the precision of the method in BPM, and is defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|HR'_i - HR_i| \tag{2}$$

Root Mean Squared Error (RMSE)

The RMSE is similar to MAE, but penalizes outlier heart rates more strongly:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(HR'_i - HR_i)^2} \tag{3}$$

Waveform Correlation (r_{wave})

The waveform correlation, r_{wave}, is the Pearson correlation coefficient (r) between the ground truth and predicted waves.

Heart Rate Correlation (r_{hr})

The heart rate correlation, r_{hr}, is the Pearson correlation coefficient (r) between the ground truth and predicted heart rates over time.

Datasets

DDPM

The DDPM dataset [7] is an interviewer interviewee deception detection dataset in which a paid actress asked 24 questions of a given subject. For each particular question subjects were instructed to either be truthful or deceptive, but otherwise were free to behave as they saw fit. It is comprised of 86 sessions of simultaneous video and pulse recordings. Sessions lasted approximately 11 minutes each. DDPM was natively recorded at 90 FPS. The original RGB videos were losslessly compressed using FFV1, a lossless intra-frame codec developed within FFmpeg.

PURE

PURE [8] is a benchmark public rPPG dataset consisting of 10 subjects recorded over 6 separate sessions of varying controlled head motions. Each session lasted approximately 1 minute. A finger pulse oximeter was used to capture the subjects' physiological signals. It was captured natively at 30 FPS. Each frame in the dataset was saved and stored as an individual png image file.

Experiments

Compression Guidelines

Each video in our two datasets was compressed with the FFmpeg libx264 implementation of H.264 at CRF values from

6 to 36, stepping by 6 [2]. We did not include the H.264 default, CRF23, but did include CRF28, the H.265 default. We then tested CRFs of 20, 21, and 22 to investigate the range in which statistical changes were evident. For H.265 we tested the same CRF values as H.264 since the bitrate still changes exponentially. We used the default encoding speed of ”medium” and the default YUV420 pixel format [1]. The audio stream was ignored.

For VP9 we sought to select CRF parameters such that the bitrate of the produced constant quality videos closely matched the bitrate of each tested H.265 CRF level. We encoded a subset of five videos from the DDPM dataset (subjects “2020-043-059”, “2020-048-050”, “2020-048-052”, “2020-049-055”, and “2020-050-062”) at 40 different VP9 CRF parameters. For each constant quality level, the bitrates of the five videos were averaged and compared to the average bitrate across all videos of each H.265 CRF level. The following CRF values were selected to test: 8, 18, 30, 35, 37, 40, 45, 50, 52, and 59. The VP9 compression algorithm was run with the equivalent of the “medium” preset, “-cpu-used 3”, and it ignored the audio stream.

The ffmpeg commands utilized for each codec can be seen in Table 1. The variables for each command include the input and output videos, the output FPS, and the CRF.

Table 1: Generalized FFmpeg commands

Codec	FFmpeg Command
H.264	ffmpeg -nostdin -y -i {input_video} -r {fps} -c:v libx264 -crf {crf} -an {ouput_video}
H.265	ffmpeg -nostdin -y -i {input_video} -r {fps} -c:v libx265 -crf {crf} -an {ouput_video}
VP9	ffmpeg -nostdin -y -i {input_video} -cpu-used 3 -r {fps} -c:v libvpx-vp9 -b:v 0 -crf {crf} -an {ouput_video}

Train on Uncompressed; Test on Compressed

FFmpeg’s encoder implementations x264, x265, and libvpx-vp9 were used to compress our videos in accordance to the H.264, H.265, and VP9 codecs, respectively. Compression was performed using all the unique combinations of the CRF and parameters described in the ”Compression Guidelines” section for each encoder across all videos in both DDPM and PURE. This resulted in the creation of 30 uniquely compressed DDPM datasets containing 96 videos each and 30 uniquely compressed PURE datasets containing 59 videos each. In total we generated 60 uniquely compressed datasets and 4650 videos.

Since the frame rate for DDPM was recorded at 90 FPS and PURE was captured at 30 FPS, we upsampled PURE and downsampled DDPM such that we had all video arrays for both datasets stored at both 90 FPS and 30 FPS. This was done to accommodate the models trained at specific frame rates and to test if the performance effects witnessed in 3DCNNs for pulse estimation at varying frame rates were present in compressed videos [5]. Unlike the original frame rate reduction method employed in Speth et al. [5] that called for skipping frames, we utilized the same method seen in Vance et al. [17] to downsample DDPM from 90 FPS to 30 FPS. This method takes place before the videos are cropped by averaging each pixel value across sets of three frames. This change was made to better emulate the behavior seen in a camera with a slow shutter speed. To upsample the PURE video arrays from 30 FPS to 90 FPS we utilized trilinear interpolation. Together these two samplings created an additional 60 uniquely compressed datasets and brought the total number of compressed videos analyzed to 9300.

The originally recorded DDPM dataset was losslessly encoded from raw video using FFV1, the lossless intra-frame codec developed within FFmpeg. The videos resulting from this encoding were used to train RPNet via the aforementioned pipeline. We train the model using 5-fold cross-validation by splitting the dataset into five disjoint subsets, each including 77 of the originally released 96 videos. The remaining 19 videos were excluded based upon inconsistencies in the original data. For each split, one subset was used for each of the testing and validation sets and three were merged to form the training set. The training was completed for a duration of 40 epochs using the negative Pearson loss function and Adam optimizer configured with a .0001 learning rate. The best model based on the loss function from each fold was then selected to test on compressed data.

All 90 FPS DDPM and upsampled 90 FPS PURE datasets were tested against this model. A second model using the downsampled FFV1 30 FPS DDPM dataset was trained under the same parameters described above. The resulting model was used to test all the downsampled 30 FPS DDPM and 30 FPS PURE datasets. For TS-CAN the same datasets were used to create TS-CAN 90 FPS and TS-CAN 30 FPS models. The training was completed for 80 epochs using the Adam optimizer configured with a learning rate of .0001 and with a model depth of 2 [18].

Train and Test on Compressed

We set up experiments in which we trained models on each compressed DDPM and PURE dataset in their native FPS resolutions. Thus, for each dataset, codec, and tested CRF combination, there exists a model trained specifically on that uniquely compressed variant, i.e. a model was trained with every unique codec and CRF combination of the compressed DDPM 90 FPS datasets. The same holds true for every PURE 30 FPS dataset. This training was done following the same manner as our original training outlined in the ”Train on Uncompressed; Test on Compressed” section. We once again utilized disjoint splits in order to perform 5-fold cross validation. When training on PURE, the total number of videos in a reassembled split was 59. No videos were excluded from the PURE splits.

This training yielded 60 unique models. For the 30 DDPM models generated through this training regime, each one was used to test all the unique 90 FPS DDPM compressed datasets within the given codec of its training data. For example, the model generated from the H.264 compressed dataset of 90 FPS DDPM using a CRF of 12 was tested against itself and every other H.264 compressed 90 FPS DDPM dataset. Using the 30 FPS PURE models generated through this training, the same testing regime was applied to the 30 FPS PURE compressed datasets. In total each of the sixty models was used to test 10 datasets.

Results
Train on Uncompressed; Test on Compressed

A noticeable decrease across all performance metrics was observed as the amount of compression increased. This decrease was witnessed for 90 FPS DDPM, 90 FPS PURE, 30 FPS DDPM, and 30 FPS PURE. As seen in Table 2, the 90 FPS DDPM datasets dataset. All metrics are calculated with a 95% confidence interval.

Codec	CRF	ME	MAE	RMSE	r_{wave}	r_{HR}
FFV1	none	2.081 ± 1.551	4.265 ± 0.841	9.982 ± 2.431	0.541 ± 0.037	0.645 ± 0.082
H.264	6	2.106 ± 1.549	4.298 ± 0.827	10.031 ± 2.400	0.539 ± 0.037	0.639 ± 0.077
	12	1.881 ± 2.396	4.622 ± 1.458	10.435 ± 2.983	0.524 ± 0.040	0.625 ± 0.090
	18	1.605 ± 3.001	5.181 ± 2.023	11.222 ± 3.389	0.500 ± 0.042	0.591 ± 0.107
	20	1.156 ± 4.091	5.972 ± 2.686	12.186 ± 3.793	0.471 ± 0.045	0.569 ± 0.101
	21	0.670 ± 4.969	6.890 ± 3.242	13.194 ± 3.991	0.443 ± 0.047	0.530 ± 0.102
	22	-0.156 ± 6.339	8.438 ± 4.147	15.079 ± 4.708	0.400 ± 0.048	0.469 ± 0.108
	24	-2.929 ± 8.033	13.274 ± 5.107	20.607 ± 5.556	0.294 ± 0.046	0.270 ± 0.105
	28	-7.988 ± 10.098	22.572 ± 5.677	29.247 ± 5.686	0.155 ± 0.030	0.066 ± 0.055
	30	-10.354 ± 10.364	25.673 ± 5.575	31.477 ± 5.693	0.089 ± 0.026	0.032 ± 0.031
	36	-10.473 ± 10.494	27.019 ± 5.193	32.314 ± 5.408	0.028 ± 0.012	0.021 ± 0.027
H.265	6	2.067 ± 1.789	4.386 ± 0.998	10.145 ± 2.693	0.538 ± 0.038	0.639 ± 0.093
	12	1.976 ± 2.053	4.557 ± 1.142	10.363 ± 2.819	0.529 ± 0.041	0.629 ± 0.095
	18	1.509 ± 4.355	6.082 ± 2.758	12.200 ± 3.875	0.475 ± 0.051	0.570 ± 0.114
	20	0.956 ± 5.709	7.211 ± 3.623	13.406 ± 4.493	0.440 ± 0.055	0.533 ± 0.109
	21	0.710 ± 6.275	8.093 ± 3.990	14.423 ± 4.656	0.410 ± 0.057	0.495 ± 0.114
	22	-0.397 ± 7.034	9.536 ± 4.461	16.120 ± 5.154	0.373 ± 0.053	0.440 ± 0.123
	24	-2.981 ± 8.754	14.597 ± 5.828	22.198 ± 6.354	0.276 ± 0.053	0.233 ± 0.105
	28	-7.152 ± 10.719	22.795 ± 5.113	29.673 ± 5.052	0.153 ± 0.035	0.066 ± 0.028
	30	-8.316 ± 11.757	25.725 ± 5.489	31.940 ± 5.330	0.105 ± 0.027	0.038 ± 0.034
	36	-9.386 ± 12.238	28.101 ± 4.841	33.552 ± 4.965	0.019 ± 0.008	0.017 ± 0.042
VP9	8	1.970 ± 1.748	4.409 ± 0.978	10.173 ± 2.577	0.539 ± 0.037	0.640 ± 0.083
	18	2.033 ± 2.208	4.660 ± 1.295	10.414 ± 2.823	0.525 ± 0.038	0.629 ± 0.090
	30	0.793 ± 4.848	6.807 ± 3.242	12.927 ± 4.059	0.446 ± 0.049	0.543 ± 0.099
	35	0.548 ± 5.569	7.225 ± 3.883	13.408 ± 4.712	0.445 ± 0.050	0.532 ± 0.111
	37	0.283 ± 5.844	7.690 ± 4.133	13.925 ± 4.952	0.429 ± 0.051	0.513 ± 0.113
	40	0.050 ± 7.128	9.354 ± 4.817	15.970 ± 5.677	0.385 ± 0.055	0.447 ± 0.134
	45	-0.231 ± 7.210	10.557 ± 5.011	17.326 ± 6.053	0.350 ± 0.057	0.392 ± 0.150
	50	-1.768 ± 8.800	14.973 ± 5.673	22.473 ± 6.358	0.266 ± 0.053	0.224 ± 0.116
	52	-2.614 ± 9.431	17.711 ± 6.019	25.335 ± 6.297	0.223 ± 0.050	0.147 ± 0.078
	59	-3.626 ± 10.210	22.119 ± 5.050	29.131 ± 4.856	0.148 ± 0.047	0.046 ± 0.038

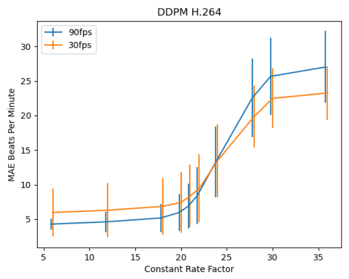
at the highest compression levels (CRF36 for H.264 and H.265 and CRF59 for VP9) are displaying a RMSE of over 29 beats per minute and a r_{HR} of less than .05. This suggests that the RPNet model trained on FFV1 90 FPS DDPM is unable to accurately provide the pulse waveform at these levels. Of particular interest in determining where RPNet breaks down is r_{HR}. For a predicted heart rate to be correlated relatively well with the ground truth, this measure should be at least .5. For H.264 and H.265 the r_{HR} drops below the .5 threshold at a CRF level of 22. For VP9 the r_{HR} drops below the .5 threshold at a CRF level above 40.

Evaluating MAE is a key metric when performing rPPG research. Figures 1 and 2 trace the behavior of MAE as the constant rate factor changes for DDPM and PURE at 30 FPS and 90 FPS. Perhaps most evident in the graphs is the much higher MAE for PURE than DDPM. The RPNet model we used was trained on the lossless FFV1 DDPM dataset. Since PURE is overall a less complicated dataset, it is not surprising the model did not perform well when testing against it. This result echoes the findings of Vance et al. [17] when they found that without augmentations models trained on DDPM and tested on PURE suffered poor performance. Despite the obviously poorer performance our models had when testing against PURE, the graphs still illustrate a rise in MAE as the CRF increases. The initial increase seen for PURE in the H.264 and VP9 graphs happens rapidly between CRF6 and CRF12 and CRF8 and CRF30, respectively. For H.265 there is a slightly more gradual increase until CRF20, at which point a sharper increase is observed. This increase is seen for both the 30 FPS and 90 FPS datasets at relatively the same rate. The trends are exclusive to the PURE datasets and suggest that a model not suited for the test dataset may quickly lose the ability to extract PPG data as the CRF increases.

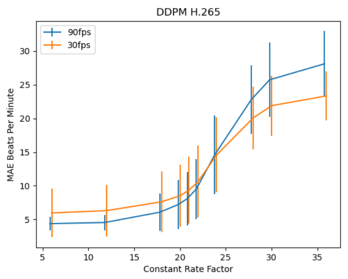
The 30 FPS and 90 FPS trained models track even closer when tested on the compressed variants of DDPM and overall have a lower MAE than when testing against PURE. An increase in MAE across the board is still witnessed as the constant rate factor is increased. This increase is somewhat gradual for the VP9 compressed datasets, with the sharpest rise beginning to take hold around CRF40. The graphs for the H.264 and H.265 codecs appear to be almost identical. Both codecs begin to suffer a steep increase in MAE between CRF20 and CRF22, which then accelerates much quicker as CRF values above 22 were used to generate the test dataset.

As stronger compression is applied, the average ability of RPNet to predict a subject's heart rate within 5 BPM falls off very quickly. The FDA allows pulse oximeters to have an error range between 2% and 3%, which roughly translates to about 5 BPM. The base value for the mean average error appears to quickly surpass these levels for each codec and CRF, but the range of any given MAE is quite large when considering the 95% confidence intervals. For instance, for the DDPM dataset compressed using H.265 and CRF21 the MAE confidence interval is ±3.99. With the base MAE of 8.093 it is possible that some videos compressed under such parameters could fall below the 5 BPM threshold.

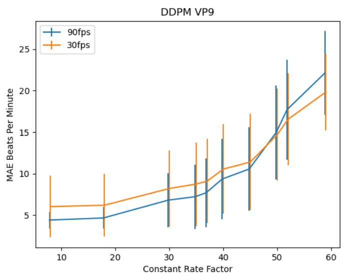
The graphs in Figure 2 further reveal the variability that large confidence intervals can have on performance. Testing on PURE results in MAE figures with very large error bars. Due to the



(a)

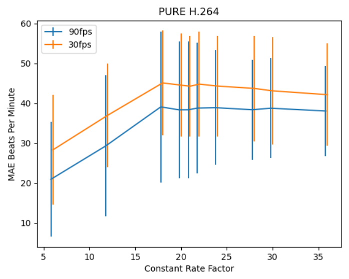


(b)

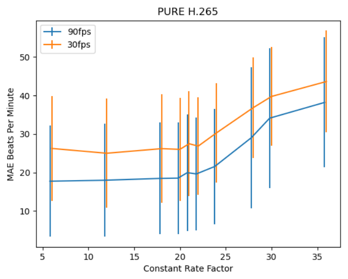


(c)

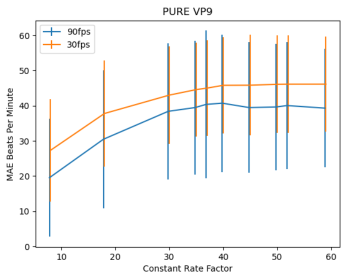
Figure 1: The change in MAE as the compression rate factor increases for H.264, H.265, and VP9 for DDPM using RPNet.



(a)

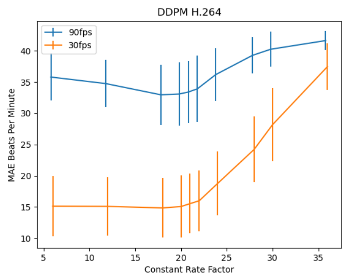


(b)

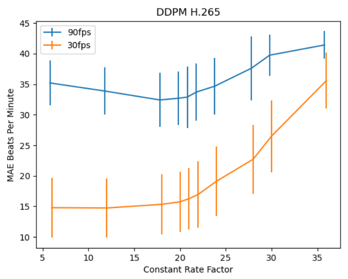


(c)

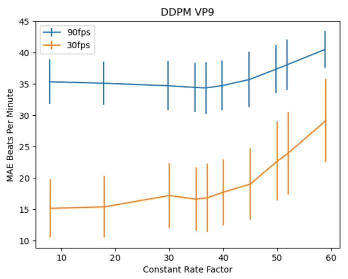
Figure 2: The change in MAE as the compression rate factor increases for H.264, H.265, and VP9 for PURE using RPNet.



(a)

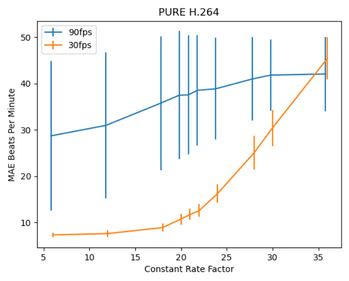


(b)

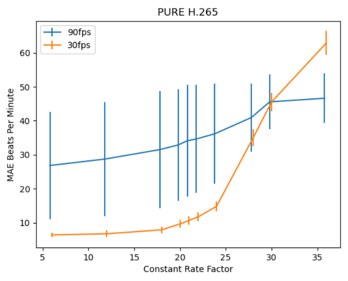


(c)

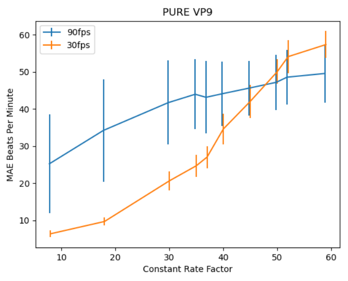
Figure 3: The change in MAE as the compression rate factor increases for H.264, H.265, and VP9 for DDPM using TS-CAN.



(a)



(b)



(c)

Figure 4: The change in MAE as the compression rate factor increases for H.264, H.265, and VP9 for PURE using TS-CAN.

inherent difficulty in cross-dataset testing between DDPM and PURE the sample mean MAE is well above the FDA threshold. Even in the presence of large error bars the testing data always results in a MAE well outside the 5 BPM range.

For the VP9 datasets seen in Figure 1c, CRF40 is the cut-off for maintaining a MAE below 5 within the 95% confidence interval. The drop below 5 is guaranteed to occur at CRF22 for H.264 and CRF24 for H.265. At lower CRF levels the 30 FPS DDPM datasets perform slightly worse than the 90 FPS ones. This suggests that videos with a higher frame rate more readily retain the rPPG data under a low amount of compression. Additionally, videos compressed with VP9 appear to better maintain PPG information than either H.264 or H.265 at higher compression levels. At the highest VP9 compression level, CRF59, the MAE is 22.119 ± 5.050. For H.264 and H.265 similar MAE of 22.572 ± 5.677 and 22.795 ± 5.113, respectively, are witnessed at CRF28, the equivalent of CRF50 in VP9.

Similar trends to those seen in the RPNet models are witnessed in the TS-CAN based models. When testing using TS-CAN models, the datasets with higher levels of compression tend to suffer performance loss. The model trained on FFV1 90 FPS DDPM videos performs poorly across all compressed datasets for both PURE and DDPM. In Figure 3, the average MAE for all codecs and compression levels is above 30 beats per minute for every compressed 90 FPS dataset. This poor performance could be the result of a lack of dilation. As noted in the "RPNet" section, RPNet utilizes temporal dilations to increase its receptive field and create an architecture more agnostic to frame rate changes. The TS-CAN architecture does not share this attribute. Additionally, since TS-CAN works with frame differences, it is possible that $1/90$th of a second delta is overwhelmed with noise and incapable of inferring the derivative of the pulse wave. The overall poor performance results seemingly mean that TS-CAN is less robust to higher frame rates than RPNet and make it challenging to say anything concrete about the behavior of the 90 FPS TS-CAN model. It should be noted that these models do see a slight improvement as the compression level of the test datasets near the default level of each codec. This observation is most apparent in Figures 3a and 3b, but even with this improvement the MAE still rests at over 30 beats per minute. These results suggest that for video captured at 90 FPS TS-CAN is unable to recover any viable PPG signal.

Diminishing performance is strongly witnessed when analyzing the behavior of the TS-CAN models trained on the down-sampled FFV1 30 FPS DDPM dataset. This trend is present across all codecs for both DDPM and PURE as seen in Figures 3 and 4. Much like RPNet, a significant decrease in performance begins to to take hold around CRF22 for both H.264 and H.265. Datasets compressed at CRFs beyond CRF24 suffer a sharp increase in performance degradation. For DDPM 30 FPS H.264 datasets, the TS-CAN model begins predicting heart rates with a MAE in excess of 25 at CRF28. This result is almost double the MAE of 12.61 ± 1.14 at CRF22 and demonstrates the sharp increase in error as CRF levels begin to move beyond the recommended default range.

When analyzing the H.265 PURE models seen in Figure 4b the sharp performance degradation is readily apparent. At CRF22 TS-CAN achieved a MAE of 11.715 ± 1.269. At CRF28 the MAE increased more than three fold from that seen at CRF22 to 35.002 ± 2.592. Between CRF6, which achieved a MAE of 6.38 ± 0.695, and CRF22 the MAE less than doubled. For VP9, the increase is more gradual. Between CRF37 and CRF40 and CRF40 and CRF45 the MAE increases by approximately 7.5, between all other sequentially tested compression levels it increases by approximately 4. Regardless of codec, the 30 FPS TS-CAN model performs worse than its 90 FPS counterpart for the highest compression level datasets.

Comparing the MAE between the TS-CAN and RPNet models shows that the TS-CAN models perform worse in all cases except for when tested against the compressed 30 FPS PURE datasets. The TS-CAN models outperform the RPNet models significantly, which could be a result of PURE being one of the datasets heavily used in the creation of TS-CAN. In the case of H.264 and H.265 the TS-CAN 30 FPS model maintains a MAE under 10 beats per minute until the tested dataset CRF surpasses 18. For VP9, the MAE of the TS-CAN 30 FPS model also maintains a MAE of under 10 beats per minutes until the tested dataset reaches a constant quality factor of 18. As the compression factor is increased for the testing datasets these MAE values increase such that by the time higher CRF values are tested the models report MAE in excess of 40 beats per minute. This once again suggests that all PPG signal data is lost beyond a certain compression level. It is of note that the 30 FPS TS-CAN models were seen to be more robust to cross-dataset training at lower CRFs than the RPNet models.

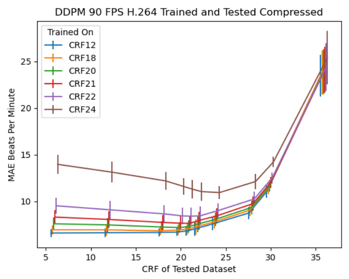
Train and Test on Compressed

When testing models trained on compressed data, we find that the effectiveness of the models is tied to the level of compression of the test set. This is in opposition to Nowara et al. [11], who found that models trained on compressed datasets performed better when tested on datasets of the same compression level. The performance of models against the test datasets can be seen in Figures 5 and 6. The performance of a given compressed dataset within each codec is plotted in relation to a model trained on a specific CRF of the same codec. The x-axis of each graph represents the CRF of the dataset that a given model is tested on.

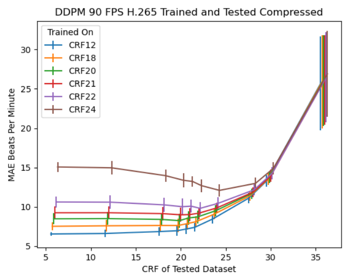
Figures 5 and 6 show that regardless of domain or codec performance steadily degrades as the amount of compression applied to the training set increases. For the 90 FPS DDPM datasets and H.264 and H.265 codecs, models trained on compressed variants between CRF12 and CRF22 experience a slight decline in MAE as the compression level of the tested dataset increases. This holds true until they are tested on datasets compressed at CRF24 or above. The models then experience a rapid increase in MAE, suggesting that beyond CRF24 PPG information is not recoverable, even for models trained on compressed data. All models experience a decrease in MAE as the compression level of the test dataset increases from CRF6 to CRF24, but the initial decrease in MAE is much greater for the model trained on CRF24. This suggests that all the models ultimately perform at their peak when tested against datasets compressed between CRF20 and CRF22. Like the other models, the one trained on CRF24 also begins to experience a rapid increase in MAE when testing on datasets compressed at CRF24 or above. All models underperformed the model trained on the FFV1 90 FPS DDPM dataset.

Examining Figure 5c demonstrates that models trained on the VP9 compressed datasets have far more disparity between performance than models trained on the H.264 and H.265 compressed datasets. The performance difference seen between models when testing the H.264 and H.265 models was very slight and in most cases almost within the 95% confidence interval of each another. This is not the case with the VP9 models, which all have much larger ranges of error. The VP9 models experience a slight decline in MAE from CRF8 to CRF40 before seeing it increase steadily through CRF59. All the compressed VP9 models were outperformed by the models trained on FFV1 90 FPS DDPM.

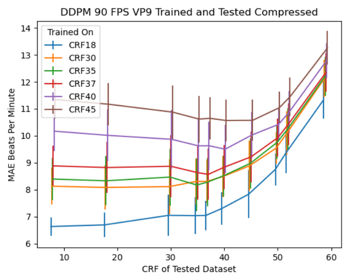
Due to the poor cross-dataset performance experienced when



(a)

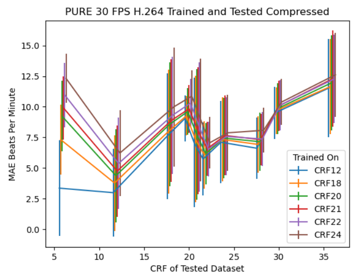


(b)

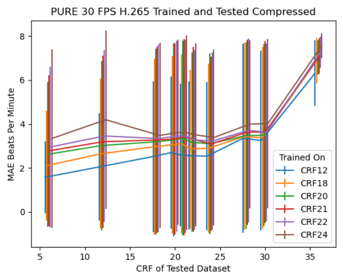


(c)

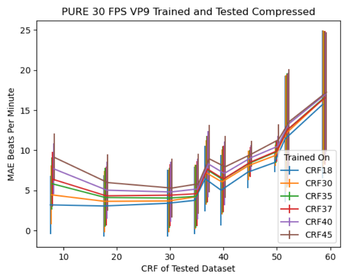
Figure 5: Figures depicting the MAE resulting from testing and training on compressed versions of 90FPS DDPM.



(a)



(b)



(c)

Figure 6: Figures depicting the MAE resulting from testing and training on compressed versions of 30FPS PURE.

training RPNet on DDPM and testing on PURE, specific PURE 30 FPS CRF models were trained. As expected, we see much stronger results than we did when using the DDPM trained model to test PURE. Viewing Figure 6 shows that the MAE for any tested dataset is relatively close between models, but increases as the CRF of the training set increases. The largest disparity occurs for models trained on less compressed data. An interesting result occurs for H.264, where MAE decreases as the CRF of the testing dataset moves from CRF6 to CRF12 and from CRF20 to CRF21 and CRF22. This suggests that even though more compression occurs at CRF12 than at CRF6 and at CRFs 21 and 22 than at CRF20 that the model is capable of overcoming the compression artifacts at theses CRF levels. For CRFs 21 and 22 this could be due to the fact that these CRFs are closer to the H.264 default of CRF23 and thus are more fine-tuned within the FFmpeg algorithm. Another possible explanation is that the videos in PURE may be more conducive to keep PPG data when compressed at these levels. VP9 also experiences a similar drop between models trained on the CRF37 dataset and those trained on the CRF40 one. The models trained and tested on the H.265 compressed datasets have the best performance. Even when testing against the CRF36 dataset the MAE is only approximately 8 BPM for all models.

Conclusions

When training is completed on uncompressed data we find that rPPG reliably breaks down once video compression reaches a constant rate factor 22 for both H.264 and H.265 and a constant quality factor (CQF) of 37 for VP9. Testing on data compressed above these thresholds begins to yield diminishing returns until it removes almost all useful PPG data at CRF36 or CQF59.

When training is performed on compressed data, we find that rPPG is not any more accurate. Regardless of codec, the most accurate model is the one trained on the lossless FFV1 version of DDPM (excluding cross-dataset training). Additionally, we find that models suffer a general degradation in performance as they are trained on compressed data of higher CRFs. This would suggest that a possible solution to the rPPG compression problem is to train models at the least compressed version possible. Models trained on low level CRFs such as CRF12 for H.264 or H.265 still perform relatively well while saving space and processing time, but not as well as those trained on lossless data. To save resources the best scenario is to evaluate losslessly-trained models on compressed data. A model trained on lossless data still performs relatively well on compressed data up until CRF22 for H.264 and H.265 or CQF37 for VP9.

Extensive studies into this domain quickly run into issues with the scalability of training models with specific compression parameters. There are many possible combinations of compression algorithm parameters one could run as just a first pass. A transcoding of any compressed video increases the number of such combinations even further. It would be quite challenging to create models such that one was trained for each possible encoding scenario and dataset. To add further difficulty to this problem, potential videos of interest may have endured multiple generations of compression, such as when videos are stored on social media sites. A video obtained from such a platform could be a result of multiple transcodings, meaning the PPG signal would essentially non-existent.

For rPPG to work effectively, we recommend processing video at compression levels below the CRF22 or CQF37 thresholds. Optimally, depending on the storage and data requirements we suggest training and testing on lossless data if possible.

References

[1] Michal Rapczynski, Philipp Werner, and Ayoub Al-Hamadi, "Effects of video encoding on camera-based heart rate estimation," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 12, pp. 3360–3370, Dec 2019.

[2] Daniel J. McDuff, Ethan B. Blackford, and Justin R. Estepp, "The impact of video compression on remote cardiac pulse measurement using imaging photoplethysmography," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, May 2017, p. 63–70.

[3] Sachin Dhawan, "A review of image compression and comparison of its algorithms," vol. 2, no. 1, 2011.

[4] G.J. Sullivan and T. Wiegand, "Video compression - from concepts to the h.264/avc standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, Jan 2005.

[5] Jeremy Speth, Nathan Vance, Patrick Flynn, Kevin Bowyer, and Adam Czajka, "Unifying frame rate and temporal dilations for improved remote pulse detection," *Computer Vision and Image Understanding*, vol. 210, pp. 103246, Sep 2021.

[6] Xin Liu, Josh Fromm, Shwetak Patel, and Daniel McDuff, "Multi-task temporal shift attention networks for on-device contactless vitals measurement," in *Advances in Neural Information Processing Systems*. 2020, vol. 33, p. 19400–19411, Curran Associates, Inc.

[7] Jeremy Speth, Nathan Vance, Adam Czajka, Kevin W. Bowyer, Diane Wright, and Patrick Flynn, "Deception detection and remote physiological monitoring: A dataset and baseline experimental results," in *2021 IEEE International Joint Conference on Biometrics (IJCB)*, Aug 2021, p. 1–8.

[8] Ronny Stricker, Steffen Müller, and Horst-Michael Gross, "Non-contact video-based pulse rate measurement on a mobile service robot," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Aug 2014, p. 1056–1062.

[9] Radim Spetlík, Jan Cech, and Jiri Matas, "Non-contact reflectance photoplethysmography: Progress, limitations, and myths," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, May 2018, p. 702–709.

[10] Zheng Zhang, Jeffrey M. Girard, Yue Wu, Xing Zhang, Peng Liu, Umur Ciftci, Shaun Canavan, Michael Reale, Andrew Horowitz, Huiyuan Yang, Jeffrey F. Cohn, Qiang Ji, and Lijun Yin, "Multimodal spontaneous emotion corpus for human behavior analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun 2016, p. 3438–3446, IEEE.

[11] Ewa Nowara and Daniel McDuff, "Combating the impact of video compression on non-contact vital sign measurement using supervised learning," 2019, p. 0–0.

[12] Weixuan Chen and Daniel McDuff, "Deepphys: Video-based physiological measurement using convolutional attention networks," 2018, p. 349–365.

[13] Justin R. Estepp, Ethan B. Blackford, and Christopher M. Meier, "Recovering pulse rate during motion artifact with a multi-imager array for non-contact imaging photoplethysmography," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2014, p. 1462–1469.

[14] Ewa M. Nowara, Daniel McDuff, and Ashok Veeraraghavan, "Systematic analysis of video-based pulse measurement from compressed videos," *Biomedical Optics Express*, vol. 12, no. 1, pp. 494, Jan 2021.

[15] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann, "Mediapipe: A framework for building perception pipelines," , no. arXiv:1906.08172, Jun 2019, arXiv:1906.08172 [cs].

[16] Gerard de Haan and Vincent Jeanne, "Robust pulse rate from chrominance-based rppg," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 10, pp. 2878–2886, Oct 2013.

[17] Nathan Vance, Jeremy Speth, Benjamin Sporrer, and Patrick Flynn, "Promoting generalization in cross-dataset remote photoplethysmography," 2023, p. 5984–5992.

[18] Nathan Vance and Patrick Flynn, "Refining remote photoplethysmography architectures using cka and empirical methods," , no. arXiv:2401.04801, Jan. 2024, arXiv:2401.04801 [cs].

[19] Xin Liu, Girish Narayanswamy, Akshay Paruchuri, Xiaoyu Zhang, Jiankai Tang, Yuzhe Zhang, Soumyadip Sengupta, Shwetak Patel, Yuntao Wang, and Daniel McDuff, "rppg-toolbox: Deep remote ppg toolbox," , no. arXiv:2210.00716, Nov. 2023, arXiv:2210.00716 [cs].